Opinnäytetyö (AMK)

Tietojenkäsittelyn koulutus

MICTTS15

2017

Kirsi Vainio

TESTAUSPOLITIIKKA JA -STRATEGIA

– Case: HealthFOX

TURKU AMK
TURKU UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖ (AMK) | TIIVISTELMÄ

TURUN AMMATTIKORKEAKOULU

Tietojenkäsittelyn koulutus

2017 | 34 + 1 liite

Kirsi Vainio

TESTAUSPOLITIIKKA JA -STRATEGIA

Case: HealthFOX

Opinnäytetyön tarkoituksena oli dokumentoida yrityksen testauspolitiikka ja -strategia koko mobiilisovellusten elinkaaren ajan sekä tutkia automaatiotestauksen mahdollisuutta. Opinnäytetyö on tehty toimeksiantona HealthFOX Oy:lle.

Teoriaosuudessa käydään läpi hyvinvointiteknologia alan testaukseen liittyvää lainsäädäntöä, eurooppalaisia ja kansainvälisiä standardeja ja niiden sisältöä sekä testausprosessin yleisiä periaatteita. Tutkimuksessa on käytetty testaukseen liittyviä kirjallisia lähteitä että omaan kokemukseen liittyviä näkökulmia. Automaatiotestausvälineissä kartoitettiin markkinoilla olevia yleisimpiä testaustyökaluja sekä testauspalveluja tarjoavia yrityksiä.

Opinnäytetyön lopputuloksena syntyi erillinen testauspolitiikka ja -strategia dokumentti sekä testaussuunnitelman että katselmointilokin mallipohjat yritykselle luovutettavaksi ja niitä ei liitetä mukaan tähän opinnäytetyöhön.

Automaatiotestauksen osalta saatiin tehtyä vain kartoitus eri välineistä sekä valittua yksi testaustyökalu alustavasti, mutta varsinaiseen pilotointiin opinnäytetyön tekemisen aikataulussa ei päästy. Testausdokumentaatiossa automaatiotestaus otettiin huomioon. Dokumentaation päivitystarve tulee olemaan ensimmäisten versioiden jälkeen tarpeellinen, koska vasta käytäntö näyttää miten prosessi toimii.

ASIASANAT:

testauspolitiikka, testausprosessi, testaussuunnitelma, testausstrategia, automaattitestaus, mobiilisovellus

BACHELOR´S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Information Technology

2017 | 34 + 1 liite

Kirsi Vainio

TESTING POLICY AND STRATEGY

Case: HealthFOX

The purpose of the thesis was to describe the commissioning company's testing policy and strategy throughout the life cycle of the mobile applications it produces and to examine the possibility of automation testing. The thesis was commissioned by HealthFOX Oy

The theoretical part of this thesis is a review of welfare legislation, European and international standards and their content, as well as the general principles of the testing process. The study uses written sources for testing and perceptions related to personal experience. The automation testing tools identify the most common testing tools and testing service companies on the market.

The final result of the thesis was the creation of the test policy and strategy documents. Another result of the thesis was the creation of the templates of the test plan and the review log for the company. However, the latter documents are not presented in this thesis, but only delivered to the company.

In the case of automation testing, only an inventory of different tools was obtained and the selected test tool was made provisionally. The piloting of the provisionally selected testing tool was not implemented. In test documentation, the automation test was taken into account. Documentation update needs will be needed after the first versions as only the policy shows how the process works.

KEYWORDS:

test policy, test process, test plan, test strategy, automation testing, mobile application

SISÄLTÖ

LIITTEET

KUVAT

TAULUKOT

KÄYTETYT LYHENTEET TAI SANASTO

Automaatiotestaus	Vähentää manuaalisen testauksen määrää ja mahdollistaa rutiininomaisen testauksen toistot, jolloin testaajat voivat keskittyä testaamaan vaikeampia testejä, joita ei ole järkevää automatisoida
Ei-toiminnallinen testaus	Täydentää toiminnallista testausta, selvittää järjestelmän suorituskykyä, tietoturvaa ja käytettävyys ominaisuuksia
Hyväksymistestaus	Virallinen testaus, jonka avulla määritetään täyttääkö järjestelmä sille asetetut hyväksyntäkriteerit, ja jonka avulla asiakas määrittää hyväksytäänkö järjestelmä
IEC	International Electrotechnical Commission
ISO/IEC 29119	Kansainvälinen yleinen testausstandardi
Integraatiotestaus	Testataan useiden komponenttien yhteistoimintaa
ISO	International Organization for Standardization, maailmanlaajuinen kansallisten standardoimisjärjestöjen (ISOn jäsenten) liitto
ISO 9001:2015	Quality management systems – Fundamentals and vocabulary
Järjestelmätestaus	Tarkistaa ja varmistaa, että ohjelmisto täyttää liiketoiminnan asettamat vaatimukset sisällyttäen myös tekniset vaatimukset
Käytettävyystestaus	Sovellukset helppokäyttöisyyden, käyttäjäkokemuksen ja käytön tehokkuuden varmistus
Lasilaatikkotestaus	Testaajalla pääsy järjestelmän koodiin, algoritmeihin ja tietokantaan
Mustalaatikkotestaus	Järjestelmää testataan ulkoisesti havaittujen toimintojen perusteella, testaajalla ei tiedä kooditason tapahtumista
Regressiotestaus	Varmistetaan, että tehdyt muutokset eivät ole rikkoneet/muuttaneet mitään aiemmin toiminutta osaa
SFS-EN ISO 13485:2016	Terveydenhuollon laitteet ja tarvikkeet (Medical devices – Quality management systems) standardi
Suorituskykytestaus	Kuormitus- ja suorituskykytesteissä varmistetaan, että asiakkaat voivat käyttää sovellusta nopeasti ja turvallisesti
Suunniteltu testaus	Testitapaukset ja testauksen työnkuku suunnitellaan etukäteen ennen testauksen alkua
Testauspolitiikka	Organisaation ohjeet ja näkemykset testaukseen

Testausprosessi	Systemaattinen prosessi testauksen eri vaiheille
Testausstrategia	Yleissuunnitelma mitä testataan, miten testataan ja miksi testataan
Testaussuunnitelma	Suunnitelma mm. testausvaiheista, testitapauksista, aikataulutuksesta, testausympäristöistä ja -välineistä, koulutustarpeesta, testaajat
Toiminnallinen testaus	Varmistetaan järjestelmän toimivuus liiketoiminnan asettamat ja tekniset vaatimukset
Tutkiva testaus	Testausta, jossa testitapauksia ei suunnitella etukäteen, vaan testaaja tekee uudet testitapaukset aikaisempien testien perusteella
Yksikkötestaus	Lähdekoodiin kuuluvien yksittäisten osien testaaminen

1 JOHDANTO

Mobiilisovellusten kehittyminen on ollut nopeaa viime vuosien aikana ja niiden laatuvaatimukset poikkeavat huomattavasti perinteisistä työpöytäsovelluksista. Mobiilisovellusten kehittäminen on nopea prosessi, jossa julkaistaan versioita tiheään tahtiin ja testaaminen voi jäädä kevyeksi.

Opinnäytetyön tarkoituksena on tehdä lain vaatimukset täyttävä testauspolitiikka ja –strategia HealthFOX Oy:lle sekä tutkia mahdollisuutta automaatiotestaukseen. HealthFOX on vuonna 2014 perustettu Salolainen hyvinvointiteknologian yritys, jonka toiminta-ajatus on digitaalinen palvelukonsepti tehostamaan kuntoutusta ja sairaudenhoidon laatua sekä lyhentämään kuntoutukseen käytettävää aikaa. Palvelussa on käytettävissä audiovisuaalisia harjoitteita, oman kuntoutuksen edistymisen seuraaminen, kommunikointi lääkärin ja fysioterapeutin kanssa sekä opastusta kuntoutuksen edistämiseen. Palvelun käyttäjiä ovat potilaat, fysioterapeutit ja lääkärit. Työntekijöitä yrityksellä on tällä hetkellä 6. Yritys sai patentin palvelukonseptille huhtikuussa 2017 ja sillä on nyt voimakas kasvu kansainvälisille markkinoille. Yrityksellä on tarve saada testauskäytännöt kirjattua dokumenttiin ja samalla parantaa laatua. Tällä hetkellä suurin panostus on ollut tuotekehityksessä ja testauksessa pääpaino on ollut yksikkötestauksessa kehittäjien toimesta.

Testauspolitiikassa ja -strategiassa otetaan huomioon SFS-EN ISO 13485 (vahvistettu 2016-03-11) standardin vaatimukset. Standardi SFS-EN ISO 13485 sisältää eurooppalaisen standardin EN ISO 13485:2016 ja on vahvistettu suomalaiseksi kansalliseksi standardiksi ja perustuu ISO 9001:2015 standardiin (Quality management systems – Fundamentals and vocabulary). Standardissa määritellään vaatimukset laadunhallintajärjestelmälle käytettäväksi organisaatioissa, joilla on tarve osoittaa kykynsä tarjota lääkinnällisiä laitteita ja niihin liittyviä palveluita, jotka säännönmukaisesti täyttävät asiakkaiden ja viranomaisten vaatimukset (SFS 13485). Samoin testauskäytännöissä otetaan huomioon terveydenhuollon laitteista ja tarvikkeista annetun lain (629/2010) vaatimukset sekä kansainvälinen yleinen testausstandardi ISO/IEC 29119. Viimeksi mainittu standardi ei ole vaatimus, ainoastaan suositus ja yrityksen koosta johtuen testaussuunnitelma on yksinkertaisempi. Kaikki standardit tukevat toiminnallista- ja ei-toiminnallista testausta, manuaalista ja automatisoitua testausta sekä suunniteltua että tutkivaa testausta.

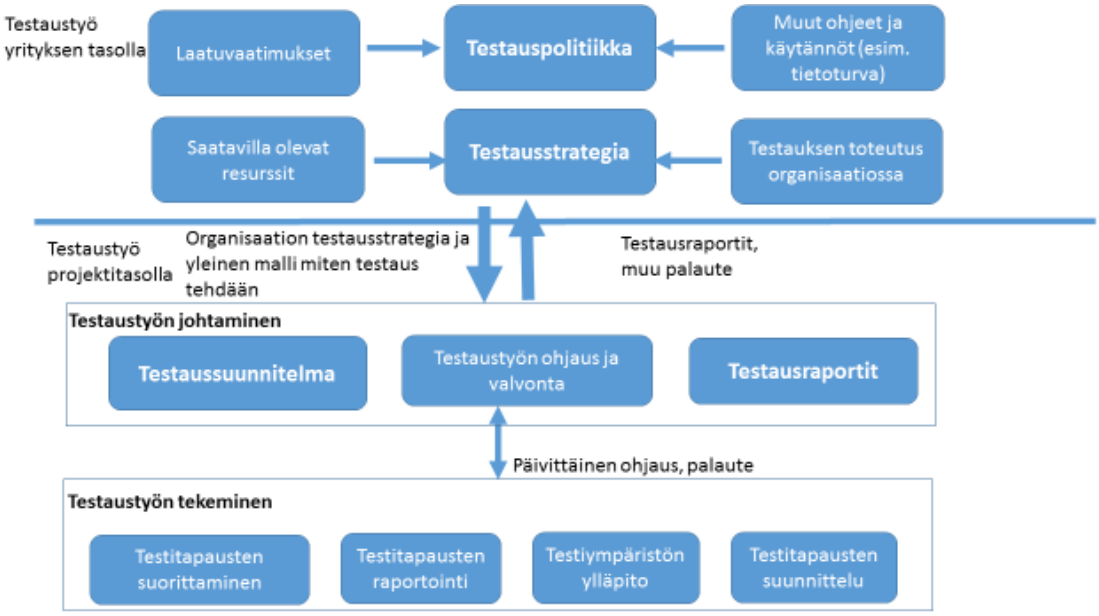
Automaatiotestausvälineiden ja testauspalveluja järjestäviin yritysten etsintään on käytetty Google Chrome, Microsoft Edge ja Mozilla Firefox –hakukoneita.

Testauksessa on nyt pääosin käytetty lasilaatikkotestausta, jossa kehittäjillä/testaajilla on suora näkyvyys sovelluksen koodin, algoritmeihin ja tietokantaan. Yritykselle tehdään ylemmän tason testauspolitiikka ja -strategiasuunnitelma, testaussuunnitelman mallipohja joka julkaisua varten sekä katselmointilokin mallipohja, jota voi käyttää kaikissa katselmointitilanteissa. Mahdollisessa tulevassa automaatiotestauksessa katselmointilokia ei tarvitse käyttää vaan käytetään testausvälineen omaa lokia. Testausdokumentaatio tulee olemaan osa yrityksen laatukäsikirjassa olevaa laadunhallintajärjestelmää.

2 TESTAUSPOLITIIKKA JA -STRATEGIA

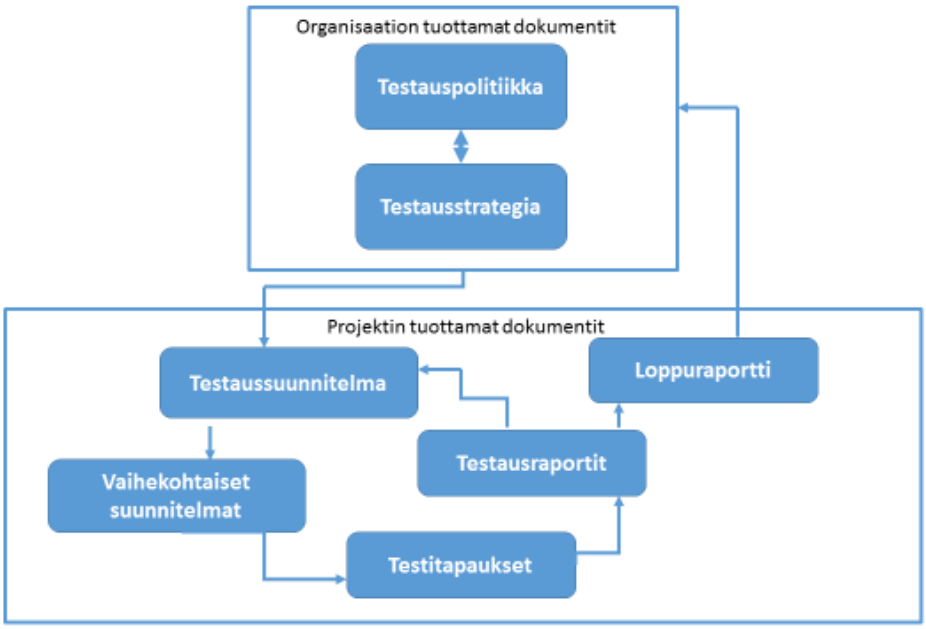
Standardissa ISO/EIC 29119 on neljä keskeistä dokumenttia: testauspolitiikka, testausstrategia, testisuunnitelma ja testausraportit (ISO/IEC/IEEE 29119). Näiden dokumenttien pohjalta määritellään ja toteutetaan koko organisaation testaustoiminta, tehdään testausta koskevat päätökset sekä tarvittaessa korjataan toiminnassa huomattuja epäkohtia. Testauspolitiikka ja testausstrategia ovat dokumentteja, jotka määrittelevät yleisellä tasolla periaatteet sille, miten testausta tehdään. Nämä dokumentit ovat ylemmän johdon suunnittelemia tai ainakin sen hyväksymiä. (Kasurinen, 2015).

Testauspolitiikassa määritellään haluttu laatu sekä tahtotila mitä testauksella halutaan saavuttaa. Testausstrategiassa kuvataan enemmän yksityiskohtia testauksen tekemiselle. Testaussuunnitelma on projektitason dokumentti, joka määritellään organisaation testausstrategian ja testauspolitiikan pohjalta. Testaussuunnitelmassa kuvataan testauksen suunnittelu ja testauksen organisointi. Testausraportti on neljäs keskeinen testauksen dokumentti. Testausraporteilla on tarkoituksena antaa tietoa yksittäisen projektin testaustoiminnan sujuvuudesta ylemmälle johdolle. Testausraportit toimivat samalla myös organisaation tiedotuskanavana, jonka pohjalta tehdään päätös siitä pitääkö testausstrategiaa muuttaa tai päivittää vastaamaan muuttuneita tarpeita tai korjaamaan toimimattomia päätöksiä. (Kasurinen 2015.) Kuvassa 1 on esitetty hyvin yleinen testausmallin prosessikuvaus (Kasurinen, 2015).



Kuva 1. ISO/IEC 29119 Yleinen testauksen malli (Kasurinen, 2015).

Perusajatus on kuvata testauksen prosessi, tuotettavat dokumentit ja testauksessa käytettävät menetelmät. Kuva 2 kertoo dokumenttien suhteet toisiinsa.



Kuva 2. Testausdokumenttien suhde toisiinsa (Kasurinen, 2015).

Asiakkaalle luovutettavaan dokumenttiin kuvataan HealthFOXin testausprosessi jossa on tarkoitus saada aikaan yhteinen ymmärrys testausprosessin kulusta sekä rooleista

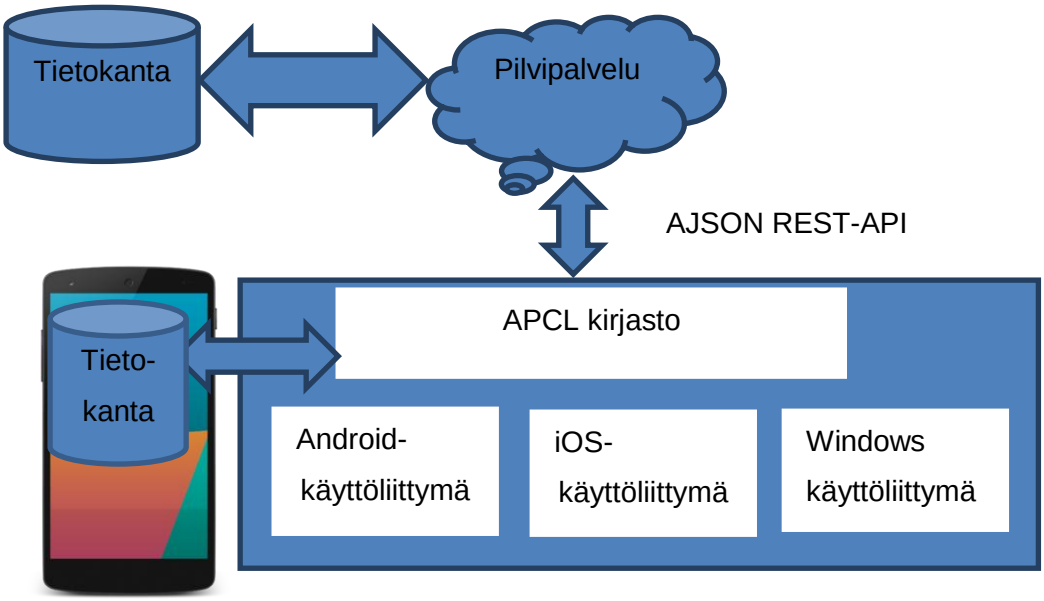
ja vastuista HealthFOX -palvelun testauksessa. Suunnitelma kattaa koko tuotteen elinkaaren, toiminnallisen ja ei-toiminnallisen testauksen. Testausprosessia kehitetään versiojulkaisuiden jälkeen koska oletus on, että vasta julkaisuiden jälkeen tullaan näkemään tarkemmin testausprosessin järkevä kulku.

Liiketoimintavaatimusten täyttäminen varmistetaan testaamalla käyttöönotettavan version järjestelmän vaatimusmäärityksissä kuvatut toiminnot.

Luotettavan toiminnan varmistaminen edellyttää virheiden systemaattista etsimistä järjestelmästä. Etsittäviä virheitä ovat loogiset virheet, koodausvirheet, kielen syntaksivirheet, tietokantojen eheysvirheet sekä muut virheet, jotka voivat tehdä järjestelmästä epävakaan. Testauksessa käytetään tyypillisten syötetietojen lisäksi myös virheellisiä ja epäkelpoja syötejoukkoja, jotta voidaan varmistaa järjestelmän toiminta odottamattomien tapahtumien sattuessa. Hyvin suunniteltu testausmenetelmä ja viimeisimpien työkalujen käyttö voivat parantaa testauksen tuottavuutta ja tehokkuutta merkittävästi (Kähkönen, 2009, 7). Virheiden korjaaminen kehityksen alkuvaiheessa maksaa vähemmän kuin niiden korjaaminen loppuvaiheessa (Dustin, 2008). Testauksessa käytetään järjestelmää varten analysoituja ja rakennettuja testitapauksia eri testaustasoille.

2.1 Kohdejärjestelmä

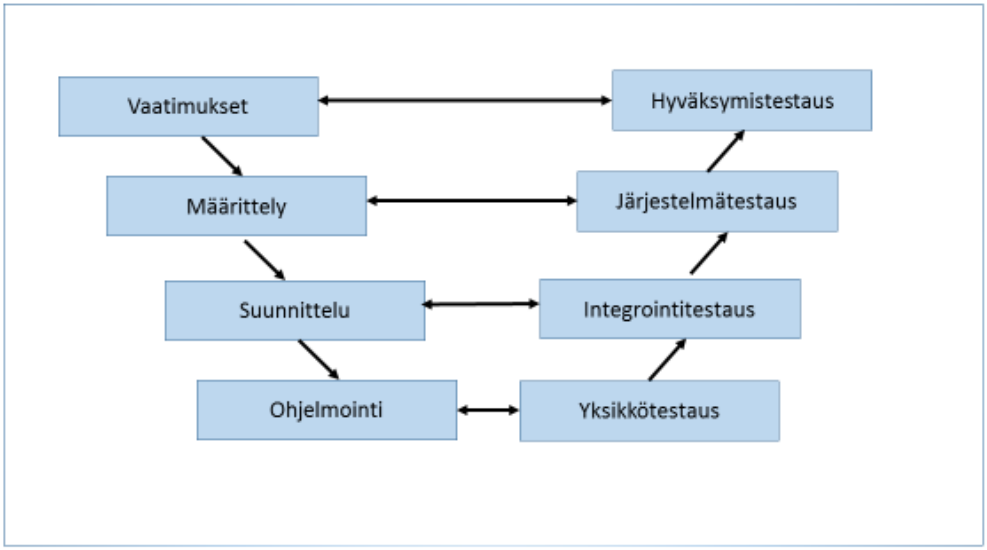
Sovelluksen kehityksessä käytetään Visual Studio 2015 kehitysympäristöä Android-, iOS- ja Windows Phone 8 alustoille käyttäen Xamarin Platform –alustaa ja MvvmCross –sovelluskirjastoa (kuva 3, Kivikoski, 2015). Xamarin Platformin alusta mahdollistaa kehittämisen C# -ohjelmointikielillä kaikilla alustoilla ja Visual Studio –kehitysympäristöllä. MvvmCross sovelluskehys helpottaa monialustasovelluksen kehittämistä MVVM-arkkitehtuuria käyttäen joka jakaa malli- ja näkymätasot kirjastoiksi erotettuna ja joita käytetään jaetusti kaikilla kolmella alustalla. Palvelun tiedot sijaitsevat pilvipalvelussa. Järjestelmän kehitys on aloitettu Windows Phone 8 pilottiversiolla vuonna 2014 ja nyt järjestelmästä on julkaistu 214 versiota. Yritys teki markkinatutkimusta ensimmäisen alustasovelluksen (Windows Phone 8) kehitysvaiheessa, nyt on käytössä enemmän asiakkaiden kanssa yhteistyö lisätarpeista (Kivikoski, 2015).



Kuva 3. HealthFOX -palvelun arkkitehtuuri (Kivikoski, 2015).

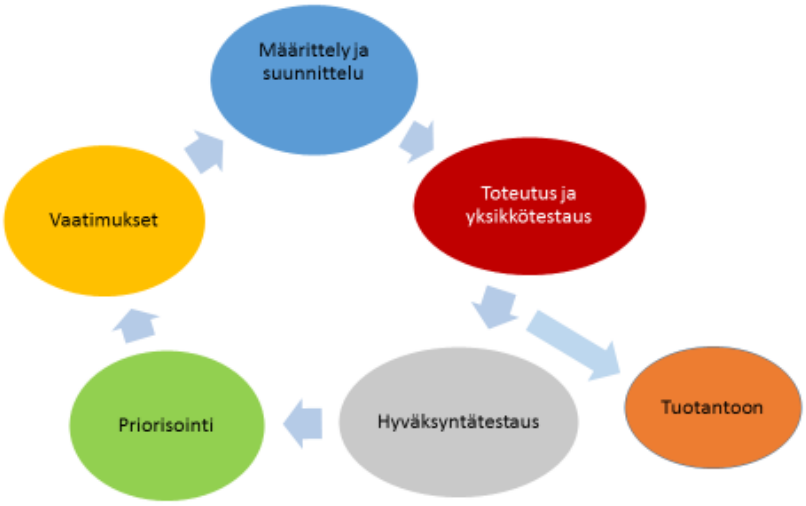
2.2 Sovelluskehitysmallit

HealthFOXin sovelluskehitys tehdään pääosin Agile menetelmin, joista käytössä Scrum. Scrumissa kehitys on jaettu lyhyisiin iteraatioihin, n. 30 päivää (kuva 5). Suuremmissa projekteissa voidaan käyttää myös klassista V-mallia (kuva 4) jossa jokaisella vaiheella on oma testausvaiheensa. Scrumissa projekti on jaettu iteraatioihin eli sprintteihin joiden lopussa on tarkoitus saada julkaisukelpoinen versio tuotantoon. Iteraatio sisältää vesiputousmallin vaiheet mutta ovat pienempiä ja nopeampia kokonaisuuksia (kuva 6).



Kuva 4. Testauksen V-malli.

Kuva 5. Scrumin kehitymalli.



Kuva 6. Scrumin vaiheet.

2.3 Testausvaiheet

Testausstrategia kattaa koko tuotteen elinkaaren ja sisältää alla olevat vaiheet sekä vesiputousmallissa että ketterässä mallissa (Scrum). Järjestelmää ovat pääasiallisesti testanneet kehittäjät toteutuksen yhteydessä, jonka jälkeen järjestelmää ovat testanneet eri henkilöt yksikkötestauksesta / betatestauksesta integraatiotestaukseen ja lopulta asiakkaan tiloissa tapahtuvaan järjestelmätestaukseen. Kaikista testausvaiheista on tehty testausraportti ja dokumentoitu yrityksen laatujärjestelmään.

2.3.1 Yksikkötestaus

Yksikkötestaus on lähdekoodiin kuuluvien yksittäisten komponenttien testaamista eli pienin mahdollinen ohjelman osa, esimerkiksi aliohjelman testaamista. Tällä varmistetaan, että ohjelman pienimmät osat toimivat kuten niiden kuuluukin ja eivät esimerkiksi kaadu jos annetaan väärä syöte tai tietokantaan lisäyksessä, poistaa tai päivittää jotain toista tietoa kannasta. Automaatiotesteillä yksikkötestauksessa on helppo todentaa ohjelmakoodiin tehtyjen muutosten vaikutukset. Agile kehityksessä automaatiotestauksen hyöty tulee vielä enemmän esille koska versioita julkaistaan usein.

2.3.2 Integraatiotestaus

Integraatiotestauksessa testataan komponenttien rajapintojen yhteensopivuutta keskenään. Tässä testausvaiheessa todetaan ohjelmistojen yhteensopivuus esim. mobiililaitteiden versioiden kanssa.

2.3.3 Betatestaus

Ohjelmiston betaversiot julkaistaan vain rajatulle ulkopuoliselle käyttäjäryhmälle, joka antaa palautetta ohjelmistosta, tällä saadaan selville ohjelmiston vikojen tai haluttujen toimintojen määrä palautteen muodossa.

2.3.4 Järjestelmätestaus

Järjestelmätestauksen tavoitteena on tarkistaa ja varmistaa, että koko ohjelmisto täyttää vaatimukset sekä liiketoiminnan että tekniseltä osalta. Testaukseen kuuluvat laitteistot, tietokannat, käyttäjät ja tietoturva ja usein myös käytettävyystestaus on mukana. HealthFOX –ohjelmistossa käytettävyys on oleellinen osa testausta. Ensimmäisissä kehitysversioissa yritys teki haastatteluja palvelun käyttäjille saadakseen realistista palautetta käytettävyydestä (Kivikoski, 2015). Järjestelmätestausvaiheessa ohjelmointivirheiden korjaaminen on yleensä aikaa vievää ja kallista. Muutoshallinta pitää olla hallittua tässä vaiheessa testausta, kaikkia muutoksia ei kannata ottaa heti tehtäväksi.

2.3.5 Suorituskykytestaus

Suorituskykytestaus (kuormitustestaus) voi olla erillinen testausvaihe tai sitten sitä voidaan tehdä samaan aikaan kuin järjestelmätestausta. Suorituskykytestauksessa pyritään selvittämään järjestelmän kyky käsitellä suurta määrää tietoa tai käyttäjiä sekä vastausaikoja. Yrityksellä suorituskyky- ja kuormitustestaus on yleensä samanaikaisesti järjestelmätestauksen aikana.

2.3.6 Regressiotestaus

Ohjelmakehityksen edetessä tulee aina enemmän ja enemmän testattavaa ja kun jotain komponenttia muutetaan, on uudelleentestauksen määrä aina isompi. Iteratiivisessa sovelluskehityksessä regressiotestauksen tarve on suuri, koska järjestelmäkehitys on nopeaa ja suuret muutokset tavallisia. Testit tulisi suorittaa usein mutta manuaalisesti tämä on mahdotonta koska kattavat testit vievät paljon aikaa. Regressiotestauksessa automaation hyöty tulee kaikkein eniten esille koska toistoja tehdään samoille testeille useita kertoja. Regressiotestauksen tarkoituksena on varmistaa, että ohjelmiin tehnyt lisäykset ja muutokset eivät aiheuta virheitä jo aiemmin toteutettujen ja toimivien komponenttien kanssa.

2.3.7 Hyväksymistestaus

Hyväksymistestaus on viimeinen testausvaihe jossa hyväksytään julkaistava versio. Testaus tapahtuu todellisessa käyttöympäristössä. Tässä testausvaiheessa virheitä ei enää saisi olla ja muutoshallinta pidetään tarkkana. Yrityksen hyväksymistestaus voidaan tehdä myös rajatulle käyttäjäkunnalle kuten betatestauskin.

2.4 Vastuut

Testausstrategiaan kuvataan testauksen organisointi ja vastuut. Testausprosessi käydään testaukseen osallistuvien henkilöiden kanssa läpi kun he ensimmäisen kerran tulevat mukaan testaukseen sekä varmistetaan testivälineiden osaaminen ja tuotettavat dokumentit. Jatkossa käydään joka version testaussuunnitelma läpi ennen testauksen alkua sekä uusien henkilöiden perehdytys. Asiakkaalle luovutettavassa testausstrategiadokumentissa on kuvattu vastuumatriisi jossa alla olevat vastuut.

Taulukko 1. Vastuumatriisi.

Vastuualue	**Vastuuhenkilö**	**Rooli**
Testausresurssien perehdytys ja koulutus (vastaa että testaukseen osallistujat tuntee testausdokumentaation sekä testauksen yleiset periaatteet)		Testauskoordinaattori

Vastuualue	Vastuuhenkilö	Rooli
Testauksen kokousten kutsuja: ○ Testauksen aloituspalaveri ○ Järjestelmätestauksen viikkopalaverit ○ Hyväksymistestauksen luovutuspalaveri ○ Tuotantoon siirron hyväksymispalaveri		Testauskoordinaattori
Vastaa testausprosessin ja testaussuunnitelman laadinnasta, testauksen etenemisen valvonnasta ja sen edistymisen raportoinnista sekä tarvittavien päätösten hakemisesta.		Testauskoordinaattori
Testaussuunnitelman hyväksyminen		Testausryhmä
Vastaa testitapausten suunnittelusta, skriptien tekemisestä ja testijoukkojen kokoamisesta		Testauskoordinaattori
Testitapausten/joukkojen katselmointi		Katselmoija
Vastaa testiympäristöjen saatavuudesta sekä testikantojen kunnosta (lähdemateriaalit ym.)		Testauskoordinaattori
Vastaa testitapausten/joukkojen suorittamisesta ja virheiden dokumentoinnista sekä testien palauttaminen alkutilaan		Testaaja/Kehittäjä
Riskien kartoitus (riskikartta liitteenä)		Testauskoordinaattori
Virheiden korjaus		Kehittäjä
Muutoshallinta testauksen aikana		Testauskoordinaattori
Testivälinetuki (mm. pääsy dokumentteihin, tietokannat)		Testauskoordinaattori
Testausraportin laatiminen ja tallentaminen hakemistoon		Testauskoordinaattori
Arvioi yhdessä testaajien kanssa testauksen lopetuskriteerien täyttymisestä ja laatii suosituksen testauksen hyväksymisestä / hyväksymättä jättämisestä		Testauskoordinaattori

Vastuualue	Vastuuhenkilö	Rooli
Testauksen hyväksyminen		Testausryhmä
Testausprosessin katselmointi		Testausryhmä

2.5 Testausprosessin katselmointi

Standardin mukaan testausprosessi on katselmoitava sovituin väliajoin (12 kuukauden välein ellei dokumenttiin ole tehty muutoksia jonkun muun syyn takia) sekä katselmoinnin tuloksena on jäätävä jälki dokumenttiin. Asiakkaalle luovutettavassa dokumentissa on hyväksyminen kohta johon katselmoija kuittaa hyväksynnän sekä käytössä on myös katselmointidokumentti johon tehdään merkintä katselmuksessa ja huomiot katselmuksesta.

Taulukko 2. Hyväksymisloki.

Versio	Päiväys	Laatija	Tarkastanut	Hyväksynyt
0.1	pp.kk.vvvv			
0.2				
1.0				

2.6 Hyväksymiskriteerit

Yritykselle kirjataan testausstrategiadokumenttiin myös ohjelmiston yleiset hyväksymiskriteerit, esim. käyttöliittymän ja serverin vasteaika, huono yhteys, virran vähäisyys. Jokaiselle julkaisulle on oma testaussuunnitelmadokumentti johon lisätään kyseiselle julkaisulle tulevat hyväksymiskriteerit. Testaus voidaan katsoa hyväksytyksi, kun kaikki testitapaukset on suoritettu, eikä avoinna ole yhtään kriittistä virhettä tai avoinna olevaan virheeseen on aikataulutettu korjaussuunnitelma. Asiakkaan testauspolitiikka ja -strategiadokumenttiin kirjataan myös vaihtoehto automatisoiduilla testitapauksille. Jos testitapauksille on laitettu prioriteetti, niin testaussuunnitelmaan kirjataan miten monta virhettä milläkin virheluokituksella voi jäädä avoimeksi.

2.7 Riskit

Yrityksen testausprosessiin tehdään riskikartta, alla esimerkinomaisesti lueteltu mahdollisia riskejä testausprosessin aikana johon lisätään vain todennäköisyys ja kokonaissumma. Testitapauksista pitäisi pyrkiä tekemään ensimmäisinä ne, jotka nähdään järjestelmän kannalta kriittisimpinä.

Vaikutuksen ja todennäköisyyden asteikko:

5 = erittäin todennäköinen
4 = korkea
3 = keskitaso
2 = matala
1 = minimaalinen

Taulukko 3. Riskikartta.

Riski nro	**Riski**	**Vaikutus**	**Todennäköisyys**	**Kok.**	**Toimenpiteet**
R1	Testaukseen jää liian vähän aikaa				Priorisoidaan testitapauksia Lisätään testausresursseja
R2	Löytyy suurempi virhe, jota ei saada korjattua				Testausryhmä harkitsee testauksen keskeyttämistä ja muutosta ohjelmistoon Julkaisun aikana tehdään katselmointeja, joilla pyritään etsimään ja korjaamaan virheet koodista ja dokumenteista ennen järjestelmätestausvaihetta
R3	Testataan vääriä asioita				Suunnitellaan testitapaukset uudelleen

R4	Testaustyökalujen käyttö ei onnistu puutteellisten taitojen suhteen				Testaus suoritetaan tarvittaessa manuaalisesti
R5	Testausympäristö ei kunnossa				Varataan aikataulu niin, että testausympäristön kuntoon laittaminen onnistuu

2.8 Koulutustarve

Testauskoordinaattorin tehtävänä on varmistaa ennen testauksen alkua, että testaukseen osallistuvat henkilöt tuntevat testausdokumentaation sekä testauksen yleiset periaatteet. Yrityksen testaus suoritetaan pääosin kahden kehittäjän työpanoksella, jotka molemmat osallistuvat testauksen vaatimiin ennakkotoimenpiteisiin jolloin testausdokumentaatio tulee olemaan tuttu heille ja koulutustarvetta ei tällä hetkellä ole.

2.9 Aikataulu

Kehittäjät testaavat samalla ajankäytöllä millä kehittävät ohjelmistoa, joten julkaisun aikataulutuksessa on otettava tarkkaan mietintään realistinen aikataulu testaukselle. Testaussuunnitelmaan tulee arviot kuhunkin testausvaiheeseen käytettävälle ajalle ja testausten jälkeen testaussuunnitelmaan päivitetään todellinen käytetty aika. Näin saadaan kokemusta kerrytettyä miten paljon kuhunkin vaiheeseen menee aikaa ja samalla voidaan kehittää prosessia poistamalla turhia kohtia ja mahdollisesti lisäämällä joitain asioita.

2.10 Kokousmenettelyt

Standardi määrittää myös minimivaatimukset kokouskäytännöille. Yrityksen testausprosessiin kirjataan vaadittavat kokouskäytännöt, käytännössä testausasiat käydään

läpi julkaisun toteutuspalavereissa. Vastuumatriisiin on kuitenkin nimetty vastuuhenkilö joka huolehtii että testausasiat tulee myös käytyä läpi.

Standardin minimivaatimukset:

- Testauksen aloituspalaveri
- Järjestelmätestauksen viikkopalaverit
- Hyväksymistestauksen luovutuspalaveri
- Tuotantoon siirron hyväksymispalaveri

2.11 Muutoshallinta testauksen aikana

Jos vaatimusmäärittelyihin tulee muutoksia, ajetaan sen vaiheen testit uudelleen ja tarvittaessa tehdään testeihin muutoksia. Testien muutostarpeen tarkistaa vastuumatriisissa nimetty henkilö. Muutoshallinta pidetään mahdollisimman hallittuna varsinkin hyväksymistestauksen aikana kuin myös järjestelmätestauksen loppuvaiheissa.

3 TESTAUKSEN VAIHEET

Alla on kuvattu testauksen käytännön työhön liittyvät vaiheet. Kuhunkin tehtävään on vastuumatriisissa nimetty vastuuhenkilö.

3.1 Testaussuunnitelman tekeminen

Yritykselle tehdään erillinen testaussuunnitelman mallipohja, johon kirjataan tulevan julkaisun tiedot ja erityispiirteet (esim. aikataulu, vastuuhenkilöt, linkki vaatimusmäärittelyihin). Testaussuunnitelma säilytetään samassa hakemistossa kuin julkaisuun liittyvät vaatimukset ja säilytysaika on standardin mukaan kaksi vuotta. Testaussuunnitelma hyväksytetään testausryhmällä johon nimetty vastuuhenkilö vastuumatriisissa. Tarkoituksena on että testaussuunnitelman tekeminen on mahdollisimman kevyt ja mallipohjaan ei tarvitse täyttää kuin julkaisun oleellisemmat tiedot. Suunnitelma sisältää tiedot julkaisusta (linkki vaatimusmäärittelyihin), testauksen aikataulut, onko testien laajuudelle jotain erityisehtoja, testauksen hyväksyntä, riskikartoitus sekä ketkä osallistuvat testaukseen.

3.2 Testien tekeminen

Testauskoordinaattori tarkistaa joka julkaisussa testitapausten ja -settien päivitystarpeen ja tekee tarvittavat muutokset ja lisäykset niihin. Testitapaukset katselmoidaan ennen testien tekemistä. Projektin aikana voidaan tehdä myös uusia testitapauksia sitä mukaa kun uusi omanaisuus on valmistunut. Testitapausten priorisointi auttaa valitsemaan ne tapaukset ensin joista on eniten haittaa järjestelmän toiminnalle. Ja silloin on helpompi tehdä testien lopettamispäätös jos aikataulu tulee vastaan ja ei ole ennätetty testata kaikkia testejä.

3.3 Testien ajaminen

Testien suorittaminen on testaajan/kehittäjän vastuulla. Yrityksen pienten resurssien takia ei ole erillisiä testaajia vaan kehittäjät suorittavat testit. Ennen testien suorittamista on testauskoordinaattori laittanut testiympäristön kuntoon sekä valmistellut testikan-

nat testauskuntoon. Testikannat palautetaan alkutilaan testaajan/kehittäjän toimesta testien ajon jälkeen.

3.4 Virheiden kirjaaminen ja raportointi

Testauksen aikana tulee esille eritasoisia huomioita, virheitä tai havaintoja jotka voivat johtua määrityksissä olleesta puutteesta, ohjelmavirheestä, väärin ymmärryksestä, laitteisto-ongelmasta tms. Dokumentaatiossa käytetään kaikista nimitystä virhe. Virheille ja virheiden tilaluokituksille tehdään neljä eri luokitusta. Samoin kirjaamiskohtaan tulee tieto korjauspäivämäärästä ja tekijästä sekä sulkemispäivä ja katselmoijan nimi.

Taulukko 4. Virheiden luokitukset.

Luokitus	**Kuvaus**	**Toimenpiteet**
Kosmeettinen	Ohjelmistossa on näytöllä kosmeettinen virhe	Virhe mahdollista jättää korjaamatta
Pieni	Ohjelmisto toimii mutta joissain tietyissä tilanteissa toimii väärin	Virhe korjataan ajan sallies-sa, testausta voidaan jatkaa ja testitapaus on suoritettu
Keskisuuri	Ohjelmisto toimii mutta tietokantaan tai näytölle menee vääriä arvoja tai muu vastaava haitta	Virhe korjataan mahdolli-simman pian, testejä voi-daan jatkaa mutta testita-paus on avoin kunnes virhe on korjattu
Kriittinen	Virheet jotka estävät ohjelmiston toi-minnan joko kokonaan tai haitta on huomattava	Virheet korjattava ennen kuin testejä voidaan jatkaa

Taulukko 5. Virheiden tilaluokitus.

Tila	Kuvaus	Vastuu
Auki	Virhettä ei ole käsitelty	Kehittäjä
Korjattu	Virhe on korjattu ja uusintatestauksessa	Testaaja
Hylätty	Virhe on hylätty (ei ollut oikea virhe)	Testauskoordinaattori
Suljettu	Virhe on korjattu ja uusintatestattu ja ok	Testaaja

3.5 Testitulosten käsittely ja hyväksyntä

Testauspalavereissa käydään läpi testausraportit ja katsotaan korjausten aikataulua sekä keskustellaan mahdollisesti virheen luokituksen muuttamisesta alhaisemmalle tasolle.

Automaattitestiajoista tulee automaattisesti testausraportti seuraavista tiedoista:

- Virheen kuvaus
- Virheen luokitus
- Virheiden määrä

Manuaalitestauksessa käytetään katselmointilokia johon virheet, luokitus ja tilatiedot kirjattu. Dokumentista nähdään myös virheiden kokonaismäärä, sekä avoinna olevat että suljetut.

3.6 Testauksen mittarit

Testauksen mittaaminen on hyvä tapa nähdä mihin suuntaan testaus on menossa sekä nähdä vertailulukuja aikaisempiin julkaisuihin. Mitattavia asioita ovat esim. testauksen läpimenoaika, testauksen työmäärä, automaation määrä, suoritettujen testien määrä ja virheiden määrät. Yritys käyttää ainakin aluksi mittarina virheiden määrää joka saadaan helposti katselmointilokista tai automaattiajojen lokista. Automaatiossa mitattavia asioita ovat esim. toteutusinvestointi, laajennusinvestointi sekä ylläpitokustannukset. Yrityksen testauksen mittarit saadaan manuaalitestauksessa katselmointilokista, joka julkaisulla on oma dokumenttinsa joten vertailua virheiden määrästä saadaan suoraan niistä. Testaussuunnitelmapohjassa on linkki testitapausdokumenttiin josta nähdään suunni-

teltujen testitapausten määrä, ajettujen testien lukumäärä ja tieto testin onnistumisesta/epäonnistumisesta.

Mittauksissa huomattava, että aina mittarit eivät ole paras tapa seurata etenemistä mutta näiden perusteella testausta voidaan kehittää eteenpäin.

3.7 Testauksen hyväksymismenettelyt

Testattava kohde voidaan siirtää seuraavalle tasolle vasta kun testaukselle asetetut vaatimukset on saavutettu. Testauspalavereissa on yhdessä sovittu mitkä virheet jätetään kyseessä olevaan julkaisuun korjaamatta ja mitkä ovat seuraavan sprintin työjonossa. Loputtomasti ei voida yrittää virheetöntä ohjelmistoa, silloin on vaarassa ylitestaus. Ohjelmissa on aina virheitä mutta hyvin testatuissa ohjelmissa virheet ovat niin ainutlaatuisia tai epätavallisia, että ne eivät haittaa ohjelman käyttöä (Kasurinen, 2015). Seuraavalle testaustasolle ei voida kuitenkaan mennä jos ohjelmistossa on kriittisiä virheitä ja niistä ei ole sovittu aikataulutettua korjaussuunnitelmaa. Yrityksen testaussuunnitelmaan kirjataan julkaisun lopetuskriteerit jos ne poikkeavat oletetusta.

3.8 Testauksen dokumentit

Testausprosessin aikana syntyvät alla olevat dokumentit jokaiselle julkaisulle.

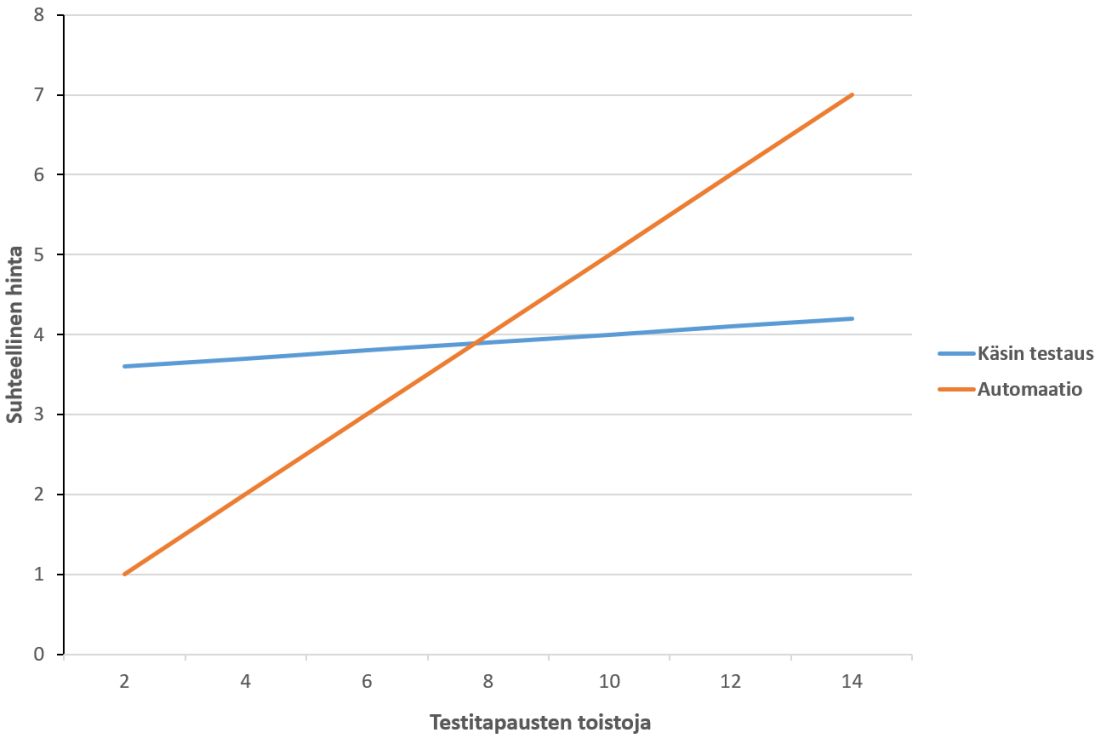
3.9 Testauksen raportointi

Joka testivaiheesta saadaan raportti joko käyttämällä yritykselle tehtyä katselmointilokia, testitapausluetteloa tai automaattitestausvälineen tuottamaa raporttia. Katselmointilokista tai automaattitestausvälineen lokista nähdään virheiden määrät, luokitukset ja tilat. Testitapausluettelosta nähdään ajettujen testien lukumäärä ja niiden onnistuminen tai epäonnistuminen. Yrityksen koosta johtuen erillistä testausraporttia ei nähdä aiheelliseksi tehdä.

4 TESTAUKSEN TYÖKALUT

Prosessin aikana tutkitaan myös automaatiotestauksen mahdollisuutta joko ostettuna palveluna tai itse käytettävää valmisohjelmistoa. Ostettavan palvelun osalta tultiin muutamien yritysten kanssa käytyjen keskustelujen jälkeen siihen päätökseen, että tässä kohtaa se ei ole ajankohtaista. Useimmat halusivat tehdä kokonaisvaltaisen testauksen ohjelmakoodin läpikäynnistä lähtien.

Automaatiotestauksella saataisiin iso hyöty tilanteissa joissa ohjelman tulee toimia kolmella eri käyttöjärjestelmällä, esimerkiksi pitäisi ajaa 15 perustestiä yhdellä käyttöjärjestelmällä eli yhteensä 45 testiä, automatisoimalla nämä, säästetään paljon aikaa. Toinen iso hyöty on suorituskyky/kuormitustestaus, jossa halutaan varmistua järjestelmän kestävyys usealla samanaikaisella käyttäjällä. Kolmas hyöty automatisoinnista on, että saadaan kokonaiskuva järjestelmän toiminnasta yhteen koottujen raporttien avulla. Toisaalta automaatiotestaus vie enemmän aikaa kuin manuaalitestaus, hyöty saavutetaan vasta toistojen myötä joka tulee esille varsinkin regressiotestauksen aikana. Useimmiten regressiotestaus on se kaikkein tylsin testausvaihe varsinkin jos ohjelmistoon on tullut useita kertoja muutoksia kehityksen aikana ja samoja kohtia pitää testata uudelleen ja uudelleen. Tutkimuksen mukaan testitapauksia jotka suoritetaan vähemmän kuin 8 kertaa muuttumattomina ei ole taloudellisesti kannattavaa automatisoida (Kasurinen, 2015). Mitä enemmän toistoja, sen halvempaa se suhteellisesti on, alla Kasurisen kustannuskäyräesitys.



Kuvio 1. Automatisoitujen testitapausten kustannuskäyrä (Kasurinen, 2015).

HealthFOX –palvelu on tehty Android, iOS ja Windows –alustoille, joten testausvälineen pitää kattaa nämä kaikki tai ainakin Android ja iOS alustat. Samoin ajatuksena, että testausväline ei ainakaan aluksi tarvitse kattaa kaikkia testausvaiheita vaan tuomaan helpotusta aikaa vieviin testeihin, esim. regressiotestausta. Selvityksessä tuli selvästi esille, että mobiililaitteiden automaatiotestaus on paljon haasteellisempaa kuin web-sovellusten. Työkalutarjonta on erittäin runsasta ja valinta edellyttää paljon selvitystyötä ja kokemusta. Monen ohjelmiston sivuilla saa äkkiä lukemalla kuvan, että juuri tämä tuote ratkaisee meidän yrityksen ongelmat sekä on helppo asentaa ja käyttää. Monen työkalun kohdalla oli mahdollista saada kokeiluversio käyttöön mutta vaati yrityksen sähköpostiosoitteen jota ei ollut nyt käytössä.

Testauspalveluiden osalta käytiin myös keskusteluja muutamien testauspalveluja tarjoavien yritysten kanssa, mm. Testimate, SGS Fimko, LogiGear ja Kilosoft. Tarkempi selvitys vaatii selvitystä testitapausten määrästä, koodien rivimääristä ym. ja karkeatakaan hinta-arviota ei pysty antamaan ilman tarkempia lukuja.

4.1 Automaattisen testauksen valmisohjelmistoja

Valmisohjelmistoista tutkittiin seuraavia:

- Team Foundation Server (TFS)
- Triecentis (Tosca Mobile+)
- Ranorex
- Origsoft
- Testarchitect
- Perfectomobile
- Calabash
- Selenium (Appium)
- Xamarin

Liitteessä 1 on kuvattu testausvälineiden eri ominaisuuksia ja alla kerrottu muutamalla sanalla joka testausvälineestä ominaisuuksista.

Team Foundation Server (TFS) soveltuu iOS, Android ja Windows –alustoille sekä on integroitavisssa yrityksen käyttämän kehitystyökalun Visual Studio 2005 kanssa. Test-Hubissa voi kirjoittaa testitapaukset ja TestExecution ja Expoloratory Testing ovat osa ilmaista pakettia. Test Manager Extension on lisämaksullinen osa, tämä sisältää mm. tarkemman raportoinnin. (TFS 2017.)

Tricentis Tosca Mobile+ soveltuu iOS (versiosta 5.0 lähtien) ja Android (versiosta 2.2. lähtien) mobiilaitteiden natiivisovelluksille. Välineellä voi testata prosessin alusta loppuun. Raportointimahdollisuuksia on useista eri näkökulmista, testitapausten määrästä, riskien kattavuudesta jne ja raportti voidaan viedä moneen eri julkaisumuotoon, esim. xml, html. Jos sopivaa raporttia ei löydy valikoimasta, voi käyttäjä tehdä oman käyttämällä Tricentis Toscan omaa kyselykieltä TQL. Tosca Mobile+ voi kokeilla maksutta 14 päivää. (Triecentis 2017.)

Ranorex soveltuu iOS ja Android testaukseen ja käyttää Seleniumin frameworkia. Ranorex tarjoaa myös versionhallinnan Git:ssä. Ranorexillä on 30 päivän ilmainen kokeiluversio, Ranorexin www-sivuilla on hyvin yksityiskohtaiset asennusohjeet. Premium lisenssi maksaa 2290 e (Ranorex 2017.)

Origsoftilla on parterina SOASTA jonka avulla voi integroida TestDrive sovelluksen toiminnallliseen ja suorituskyky testaukseen. TestDrive toimii pilvessä eikä vaadi ohjelmointitaitoja. (Origsoft 2017.)

Testarchitectissa on integrointimahdollisuus Visual Studio Team Foundation Serverin kanssa sekä monen muun sovelluksen kanssa, mm. Jira. Testarchitectissa on heidän www-sivuillaan hyvin yksityiskohtaiset asennusohjeet sekä tukipalvelut. Testausväline ei vaadi ohjelmointitaitoja, testit voidaan tehdä vedä – ja pudota –toiminnolla. Testarchitetin kautta voi ostaa myös testauspalveluja, joita LogiGear tekee.
(Testarchitect 2017.)

Perfectomobile soveltuu iOs, Android ja Windows –alustoille. Se on yhteistyössä Microsoftin Visual Studion kanssa jolloin Visual Studion käyttäjät voivat testata pilvipohjaisessa laatu laboratoriossa. Kehittäjät ja testaajat voivat kehittää automaatiota Seleniumissa tai Appiumissa C# koodilla ja ajaa esim. regressiotestejä monissa laitteissa rinnakkain. Testiskriptit ovat itsenäisiä sovelluksia ja ne voidaan integroida esim. Visual Studio Team Serviceen tai Team Foundation Serveriin. (Perfectomobile 2017.)

Calabash testaustyökalulla voi testata mobiililaitteiden (iOS ja Android) natiivisovelluksia. Xamarin ylläpitämä Calabash on avointa koodia ja ilmainen. Calabash tukee Cucumberia jolla saa dokumentaatiota enemmän liiketoiminnan ymmärtämään muotoon. (Calabash 2017.)

Xamarin on pilvipalveluihin soveltuva testaustyökalu iOs (versiosta 5.0 lähtien) ja Android (versiosta 4.3 lähtien) sovelluksille. Xamarissa voi nauhoittaa testitapaukset suoraan Visual Studiosta C# kielellä. Xamarin www-sivuilla on yksityiskohtaiset ohjeet nauhoitukseen ja testaukseen. (Xamarin 2017.)

Seleniumin Appium voi testata mobiilaitteiden (iOS ja Android) natiivisovelluksia. Jos testataan Android versiota 4.2. vanhempaa, pitää käyttää Selendroidia. Testaukseen vaaditaan oma Appium-serveri. (Selenium 2017.)

Yrityksellä on käytössä Microsoftin Team Foundation Server versionhallinnassa ja todettiin yrityksen vastuuhenkilöiden kanssa käytyjen keskustelujen jälkeen, että tutkitaan tarkemmin Team Foundation Serverin käyttömahdollisuutta automaatiotestauksessa.

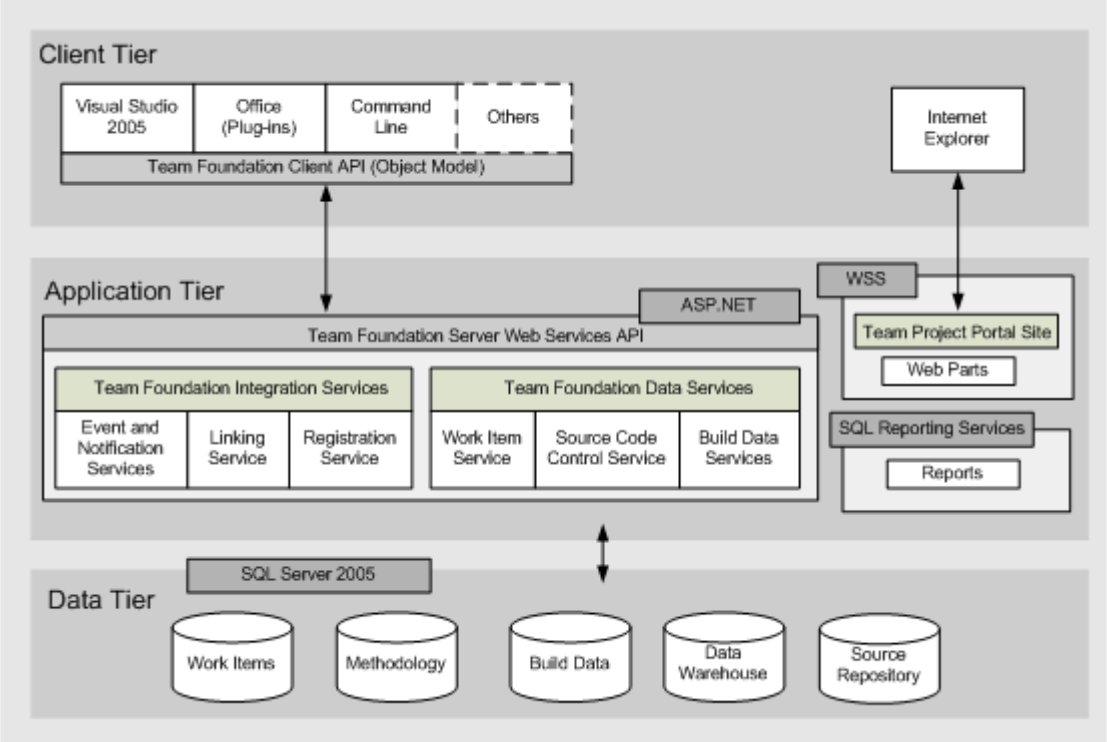
4.2 Automatisoinnin odotukset

Automatisoinnin yleisiä ongelmia ovat mm. epärealistiset odotukset, väärien asioiden testaaminen, väärä turvallisuudentunne ja uusien virheiden löytyminen. Henkilöstö voi olettaa, että saadaan paljon tuloksia pienellä työmäärällä ja automatisointiin ei varata riittävästi resursseja. Usein myös ajatellaan, että testausautomaatio vähentää automaattisesti kustannuksia kun taas tosiasiassa varsinkin alussa automaatio kasvattaa kustannuksia. Testejä automatisoidessa löytyy usein virheitä, ei niinkään niitä ajettaessa. Testejä ajettaessa ohjelmisto toimii hyvin niillä testeillä joita tehtiin mutta se ei takaa virheettömyyttä kuten ei testaus muutoinkaan. Testitapaukset vaativat myös jatkuvaa päivittämistä ja sana automaatio voi johtaa harhaan samoin kuin kokemuksen puute tuo mukanaan omat haasteensa. (Pohjolainen, P. 2003.)

Yrityksen kohdalla isoin haaste on resurssien vähyys. Automatisoinnin pilotoinnin suunnittelu ja toteutus vaativat kuitenkin kehittäjien panostusta joka on pois tuotekehityksestä. Onnistunut automatisointi vapauttaa resursseja muualle, lyhentää testauksen läpimenoaikoja sekä pienentää testauksen kustannuksia (Pyhäjärvi & Pöyhönen 2004, 16). Automatisointi ei ole vain kertaluonteinen asia vaan testien suorittamisen jälkeen analysoidaan saavutetut tulokset ja kehitetään prosessia sen mukaan.

4.3 Team Foundation Server

Team Foundation Serverin arkkitehtuuri on kolmekerroksinen sisältäen client, application ja data kerrokset (kuva 7, TFS 2017).



Kuva 7. Team Foundation Server arkkitehtuuri (TFS 2017).

4.4 Käyttöönotto

Suositeltava tapa on tehdä ensin pilotointi yksikkötestaukseen muutamalla testitapauksella jonka aikana samalla opetellaan työkalua ja nähdään miten väline toimii käytännössä. Käyttöönotto on hyvä tehdä askeleittain ja parannella käytäntöjä koko ajan. Team Foundation Server sivuilla on hyvät infosivut välineen käyttöönotolle (TFS Guide). Ensimmäiseksi vaiheeksi kannattaa ottaa muutama perustestitapaus joita voi suorittaa eri testausvaiheissa, ei missään nimessä yritetä automatisoida kaikkea.

5 YHTEENVETO

Testaukseen liittyviä standardeja on paljon joista yritys voi ottaa käyttöönsä niistä itselleen parhaiten sopivat. Kaikkia standardin prosesseja ei välttämättä tarvitse ottaa käyttöön.

Tavoitteena oli tehdä mahdollisimman kevyt ja helposti toteutettava testausprosessi lain ja standardien vaatimusten puitteissa liitettäväksi yrityksen laatukäsikirjaan sekä tutkia automaatiotestauksen mahdollisuutta.

Yritys on panostanut alkuvuosinaan voimakkaasti tuotekehitykseen ja testaus tehdään sisäisin voimin. Jatkossa pitää panostaa myös automatisoituihin testausprosesseihin, että palvelusta saadaan entistä laadukkaampi. Standardien mukainen toimintatapa auttaa myös markkinoinnissa saavuttamaan asiakaskunnan luottamuksen asiantuntijamaisella testauksella. Yritys on kasvamassa myös kansainväliseen suuntaan joten standardien mukainen testaustapa edesauttaa myös siinä, että puhutaan samoilla termeillä ulkomaisten asiakkaiden kanssa testauksen osalta.

Yritykselle saatiin testauspolitiikka ja –strategia kirjattua dokumenttiin sekä julkaisuille testaussuunnitelman ja katselmointilokin mallipohjat. Testauspolitiikan ja –strategian kehittäminen ja päivittäminen on aluksi tarpeen jokaisen julkaisun jälkeen, että päästään käytännölliseen ja konkreettiseen testausprosessiin varsinkin kun prosessia ei päästy soveltamaan vielä käytännössä.

Automaattisen testauksen suhteen on vaatimus, että testien pitää olla helposti ajettavissa ja tutkittavissa testaajien ja kehittäjien toimesta. Automaatiotestausvälineitä on markkinoilla paljon ja monet hyvinkin samankaltaisia. Pienen yrityksen ajankäytön vuoksi on hyvä vielä miettiä kannattaako yrityksen panostaa testauksen automatisointiin vai onko kuitenkin järkevämpää ostaa kokonaispalvelu ulkopuoliselta. Tosin myös ulkoistamiseenkin tarvitaan omien työntekijöiden panostusta. Työmäärä automatisointiin tulee näkyviin kun yritys tekee myöhemmin pilotin yksikkötestaukseen parille usein toistettavalle testitapaukselle.

LÄHTEET

Calabash 2017. Viitattu 6.3.2017. http://calaba.sh/

Dustin, E., Rashka, J. & Paul, J. (2008). Automated Software Testing – Indroduction, Management, and Performance (13. painos).

Hirsjärvi S, Remes P & Sajavaara P, Tutki ja kirjoita, 2009, 464 s.

ISO/IEC/IEEE 29119, Software and systems engineering – Software testing

Kasurinen, J. 2015. Ohjelmistotestauksen käsikirja. E-kirja.

Kähkönen, T. 2009. Yritysten valmius soveltaa uusia ohjelmistotuotteiden testaus- ja laatustandardeja (ISO/IEC 29119 ja 25010). Kandidaatintyö. Tietotekniikan osasto. Lappeenranta: Lappeenrannan teknillinen yliopisto.

Kivikoski, A. 2015. Alustariippumattoman sovelluksen kehittäminen kuntoutumisen työkaluksi. Opinnäytetyö. Tietotekniikan koulutusohjelma. Turku: Turun ammattikorkeakoulu.

Origosoft 2017. Viitattu 6.3.2017. https://www.origsoft.com/products/testdrive/

Perfectomobile 2017. Viitattu 6.3.2017. https://www.perfectomobile.com/solutions/perfecto-test-automation

Pohjolainen, P. 2003. Ohjelmiston testauksen automatisointi. Viitattu 3.7.2017. http://cs.uef.fi/uku/tutkimus/Teho/PenttiPohjolainen_Gradu.pdf

Pyhäjärvi, M. & Pöyhönen, E. 2004. Testausvälineet ja testauksen automatisointi. Viitattu 2.6.2017.
http://users.jyu.fi/~kolli/testaus2006/materiaali/8_TestausvalineetJaTestauksenAutomatisointi_v0_1.ppt

Ranorex 2017. Viitattu 6.3.2017. https://www.ranorex.com/

Selenium 2017. Viitattu 6.3.2017. http://www.seleniumhq.org/

SFS 13485. Terveydenhuollon laitteet ja tarvikkeet. Helsinki: Suomen standardoimisliitto.

TFS Guide. Viitattu 2.6.2017. http://www.tfsguide.com

TFS 2017. Viitattu 2.6.2017. https://msdn.microsoft.com/en-us/library/bb668952.aspx

Testarchitect 2017. Viitattu 6.3.2017. http://testarchitect.logigear.com/product/features-and-supported-platforms

Triecentis 2017. Viitattu 6.3.2017. https://www.tricentis.com/tricentis-tosca-testsuite/tosca-mobile-plus/

Testivälinevertailutaulukko

Nimi	Pilvipalvelu	iOS	Android	Windows Phone 8	Kustannusarvio	Useita raportointimahdollisuuksia	Agile + V-malli kehitykseen	Ohjelmointitaitoja vaativa	Nauhoitusmahdollisuus	Testien ajastamismahdollisuus	Manuaali- ja automaattitestaus Framework	Skriptauskieli/tuki	Integrointi Visual Studio	Linkki	Arvio toteutuksesta
TFS	Kyllä	Kyllä	Kyllä	Kyllä	Ilmainen jos Visual Studio	Kyllä	Kyllä	Kyllä	Kyllä	Kyllä	Kyllä	Microsoft.NET	C#, VB.NET	Kyllä	m/testing-in-tfs-team-services/	Helpohko
Tosca Mobile+	Kyllä	Kyllä	Kyllä	Ei	Ei tiedossa	Kyllä	Kyllä	Ei	Kyllä	Ei tiedossa	Kyllä	Useita	C#,HTML,JavaScript,Java,C++	Kyllä	https://www.tricentis.com/tricentis-tosca-testsuite/tosca-mobile-	Helpohko
Ranorex	Kyllä	Kyllä	Kyllä	Ei	2290-3900 e tai pelkkä testien runtime 690e	Kyllä	Kyllä	Ei	Kyllä	Kyllä	Kyllä	Useita	C#, VB.NET	Kyllä	com/automate-testing-of-desktop-web-mobile-software.html	Helpohko
Origsoft (TestDrive)	Kyllä	Kyllä	Kyllä	Kyllä	Ei tiedossa	Kyllä	Kyllä	Ei	Kyllä	Kyllä	Kyllä	Useita	C#,HTML,JavaScript,Java,C++	Kyllä	https://www.origsoft.com/solutions/mobile-testing/	Helpohko
Testarchhitect	Kyllä	Kyllä	Kyllä	Ei	Ei tiedossa	Ei	Kyllä	Ei	Kyllä	Ei tiedossa	Kyllä	Microsoft.NET, Selenium WebDriver	C#,Java,Python	Kyllä	ect.com/OnlineHelp/?_ga=2.2999032.976706089.1499434601-	Helpohko
Perfectomobile	Kyllä	Kyllä	Kyllä	Kyllä	299$/month /user -->	Kyllä	Kyllä	Kyllä	Kyllä	Ei tiedossa	Kyllä	Useita	C#, Java,Python,Ruby,JavaScript,Perl,ObjectiveC/Swift	Kyllä	https://www.perfectomobile.com/solutions/perfecto-test-automation	Helpohko
Selenium / Appium	Kyllä	Kyllä	Kyllä	Ei	Ilmainen	Ei	Kyllä	Kyllä	Ei	Ei tiedossa	Kyllä	Selenium / Appium	C#, Java,Python,Ruby,JavaScript,Perl,ObjectiveC/Swift	Kyllä	http://www.seleniumhq.org/download/	Ehkä työläs, vaatii mm. oman Appium -serverin. Tulosten tarkastelu kankeaa.
Calabash	Kyllä	Kyllä	Kyllä	Ei	Ilmainen	Kyllä	Kyllä	Hieman	Ei	Kyllä	Kyllä	Useita (Appium,Calabash,Xamarin.UITest)	C#, Ruby,ObjectiveC/Swift	Kyllä	http://calaba.sh/	Helpohko
Xamarin Test Cloud	Kyllä	Kyllä	Kyllä	Ei	379-799/month	Kyllä	Kyllä	Kyllä	Kyllä	Ei	Kyllä	Useita	C#	Kyllä	https://www.xamarin.com/test-cloud	Helpohko